华南理工大学

South China University of Technology

本科毕业设计（论文）外文翻译

学　　院：	机械与汽车工程学院
专　　业：	机械工程
学生姓名：	魏伟和
学生学号：	201430031255
指导教师：	张东

中文译名	二维激光雷达 SLAM 中的实时环路闭合
外文原文名	**Real-Time Loop Closure in 2D LIDAR SLAM**
外文原文版出处	**IEEE International Conference on Robotics and Automation. IEEE, 2016:1271-1278.**

二维激光雷达 SLAM 中的实时环路闭合

Wolfgang Hess , Damon Kohler , Holger Rapp 1 , Daniel Andor

摘 要：便携式激光测距仪，又称为激光雷达，同时定位和地图绘制（SLAM）是一种有效的用来获取完工建筑平面图的方法。实时生成和可视化建筑平面图有助于操作员评估捕获数据的质量和覆盖率。建立一个便携式的捕获平台需要在有限计算资源下运行。我们提出了新的的方法，用于我们的后端映射平台上，使之实现了实时建图和在分辨率为 5cm 的闭环。为了实现实时闭环，我们使用分支定界法来计算扫描子匹配作为约束。我们提供实验结果来与其他已知的著名方法进行比较，结果表明，我们的方法在质量方面与现有技术具有一定的竞争力。

1. 引言

完工的平面图对各种应用都很有用。手动调查收集建筑物管理数据任务通常需要计算机辅助设计（CAD）与激光卷尺。这些方法很慢，而且通过雇佣对建筑带有先入之见的人类来收集直线，并不能总是准确地描述真实空间的本质。通过使用 SLAM，可以快速且准确地测量建筑物的大小和手工测量无法应对的大尺度的复杂性，

在这一领域中应用 SLAM 并不是一个新概念，但是这不是本文的重点。相反，我们将提供一种减少计算量的新方法来计算环路闭合约束所要求激光测距数据。这种技术使我们能够绘制非常大尺度（大数万平方米）的平面地图，同时为操作员提供完全实时优化的结果。

2. 相关工作

扫描到扫描的匹配经常用于激光 SLAM 中的相对姿态变化计算，如[1]-[4]。然而，它本身会迅速地积累错误。

扫描到地图的匹配有助于限制这种误差的积累。有这样的一个方法，通过使用 Gauss Newton 找到线性插值映射的局部最优解，如例[5]。通过使用一个足够高的数据扫描速率的激光雷达提供，确保存在良好的初始位姿估计，那局部优化的扫描到地图的匹配将是有效的和健壮的。在不稳定的平台上，激光风扇被投射到使用了惯性测量单元（IMU）的水平面以估计重力方向。

像素精确扫描匹配方法，如例[1]，可进一步减少局部误差积累。虽然计算方面代价更高，这种方法对闭环检测也有用。一些方法侧重于通过匹配从激光扫描提取的特征来降低计算代价[4]。检测环路闭合的其他方法包括基于直方图的匹配[6]，从扫描数据进行特征检测，利用机器学习[7]。

两种常见的用来表达残余误差积累的方法为粒子滤波和基于图优化的 SLAM。

粒子过滤器必须保持能完整表示每个粒子的系统状态的表述。对于基于网格的 SLAM，这种表述随着地图变大，很快就会变得资源密集；例如，我们的一个测试用例是 22000 平方米、超过 3 公里的轨道。.较小的维度特征表示，例如[9]，并不需要为每个粒

子制备网格地图，因此可以用于减少计算资源的需求。当要求最新的网格地图时，[10]表明计算子图只有在必要时才更新，以便最终的地图是所有子图的栅格化。

基于图的方法彻底改变了关于表示姿势和特征的节点的集合。图的边是从观察中产生的约束。各种优化方法可用于将引入的有全部约束的误差最小化，例如[11]，[12]。这种户外的系统 SLAM 使用基于图形的方法、局部扫描匹配方法，和重叠的、描述在[13]中的基于子图特征直方图局部映射的匹配方法。

3. 系统回顾

关于室内建图，谷歌的 Cartographer 提供了一个实时的解决方案，用安装有传感器的背包生成r = 5cm 分辨率的二维网格的地图。当这个系统的机器人在穿过建筑物时，它可以看到正在创建的地图。激光扫描被插入到在最佳的估计位置的子图，被认为在短时间内是足够准确的。扫描匹配在最近的一个子图进行，所以它只依赖于最近的扫描。关于现实世界的位姿估计误差也在积累。

为了能够以适中的硬件来获得良好的性能，我们的 SLAM 方法不使用粒子滤波器。为了应付误差的累积，我们有规律地运行姿态优化。当一个子图完成后，将不再有有新的扫描插入它了，它将参与闭环的扫描匹配。所有完成的子图和扫描会自动被纳入到闭环的检测。如果他们在当前位姿估计下足够近似，那么匹配器将会试图在子图找到对应的扫描。如果在当前位姿的搜索窗口中找到了一个足够好的匹配，那么它将作为一个闭环约束添加到优化问题。通过每几秒钟完成一次优化，操作者得到的经验就是当一个位置被重新访问时，闭环将很快发生。这导致一个软实时约束，对闭环扫描的匹配必须比新的扫描更快速，否则，它将明显落后。我们通过使用一个分支定界法和几个预先计算的每个子图都已完成的网格来实现这一点。

4. 局部 2D SLAM

关于 2D SLAM，我们的系统结合了单独的局部和全局的的方法。这两种方法都优化了位姿$\xi=(\xi_x,\xi_y,\xi_\theta)$，它包含了一个由激光雷达观测到的平移$(x,y)$和旋转$\xi_\theta$，这也称为扫描。在一个不平坦的平台上，比如我们的背包，IMU 用来估计重力的方向，将水平扫描激光雷达投射到二维世界。

在我们的局部方法中，每个连续扫描都与一小部分的称为子图M 的现实的匹配，通过使用一种非线性优化，将扫描与子图结合起来，这个过程也被称为扫描匹配。扫描匹配会随着时间的推移累积错误，这个错误会在第五部分描述的全局方法中删除。

A.扫描

子图建设是一个迭代反复的过程，它将匹配扫描和子地图坐标框架，进一步提帧。假设初始扫描为$0\in \mathrm{R}^2$，现在我们把扫描点的信息记为$H=\{h_k\}_{k=1,\ldots,K}, h_k\in \mathrm{R}^2$。在子图中的扫描架的姿势$\xi$表示为变换$T_\xi$，这严格地将扫描点从扫描帧转换为子图的框架，定义如下式子：

$$T_\xi p = \underbrace{\begin{pmatrix} \cos\xi_\theta & -\sin\xi_\theta \\ \sin\xi_\theta & \cos\xi_\theta \end{pmatrix}}_{R_\xi} p + \underbrace{\begin{pmatrix} \xi_x \\ \xi_y \end{pmatrix}}_{t_\xi} \quad (1)$$

B.子图

部分连贯的扫描可以被用来构建子图。这些子图以概率网格 $M : rZ \times rZ \rightarrow [p_{\min}, p_{\max}]$ 的形式表示，这些概率网格是由从扫描匹配得到的给定分辨率为 r=5cm 的离散点组成的。这些值可以是关于网格点被阻塞的可能性的思考。对于每个网格点，我们定义相关的像素为最接近网格点的所有点的集合。

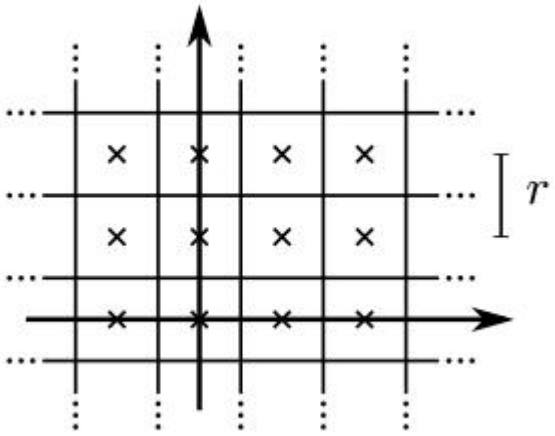

图 1.网格点和关联的像素

每当将扫描插入到概率网格中时，一组击中点和不相交集的网格点将会被计算。对于每个命中，我们将最接近的网格点插入命中集。对于每一个缺失，我们将插入一些网格点。这些网格点与像素相关联，每个像素相交于在扫描原点和每个扫描点之间的光线中的一条，不包括已经命中的网格点。以前从未观测过的网格点，如果位于一个命中集，则赋予它一个概率值 p_{hit} 。如果位于一个缺失集，则赋予它一个概率值 p_{miss}。如果网格点 x 已经被观察到，我们分别更新击中和错过的几率为

$$\text{odds}(p) = \frac{p}{1-p}, \quad (2)$$

$$M_{new}(x) = \text{clamp}(\text{odds}^{-1}(\text{odds}(M_{old}(x)) \bullet \text{odds}(p_{hit}))) \quad (3)$$

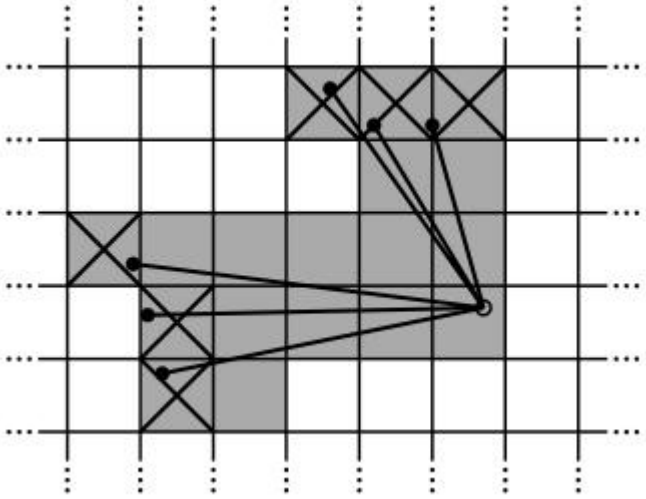

图 2.一次扫描的范围，黑色点表示击中点

C.Ceres 扫描匹配

插入扫描到一个区块之前，会根据当前局部子图，使用 Ceres 子图扫描匹配来优化扫描位姿 ξ。扫描匹配负责找到一个扫描位姿来把在子图中的扫描点的概率最大化。我们把这个过程转换为非线性最小值问题

$$\underset{\xi}{\operatorname{argmin}} \sum_{K=1}^{K} (1 - M_{smooth}(T_{\xi} h_k))^2 \qquad (\mathrm{CS})$$

其中 T_{ξ} 根据扫描位姿把 h_k 从扫描帧变换到子图帧。函数 $M_{smooth} : \mathrm{R}^2 \rightarrow \mathrm{R}$ 是局部子图概率值光滑的表述。我们使用的双三次插值。因此,值可以发生在区间[0,1]，但被认为没有危害。

通常这个平滑函数的数学优化能够给出比网格分辨率更好的精度。因为这只是一个局部优化，所以需要一个很好的初始估计。能够测量角速度的 IMU 被用来估计旋转组件扫描之间的转角 θ。更高频率的扫描匹配或像素精确扫描匹配方法，虽然带来更大的计算压力，但可以在没有 IMU 的情况下使用。

5. 闭环

当扫描只与子图包含最近的几次扫描匹配时，上面描述的方法会缓慢地积累误差。只有几十次连续扫描，累积误差较小。

较大的空间是通过创建许多小子图。.我们的方法依据稀疏的姿态调整[2]来优化所有扫描和子图的姿势。插入了扫描的相对位姿被存储在内存中以用于闭环的优化。

除了这些相对位姿，一旦子图不再变化，其他一切包含扫描和子图的都将会纳入到回环检测的考虑中。.扫描匹配是在后台运行，如果找到一个好的匹配，相应的相对姿态会被添加到优化问题中去。

A. 优化问题

闭环优化，如扫描匹配，也被表述为非线性最小二乘法问题，允许很容易地添加残差来考虑额外的数据。每隔几秒钟，我们就使用 Ceres [14]来计算下面方程的解：

$$\underset{\Xi^m,\Xi^n}{\operatorname{argmin}} \frac{1}{2}\sum_{ij}\rho\left(E^2\left(\xi_i^m,\xi_j^s;\Sigma_{ij},\xi_{ij}\right)\right) \qquad \text{SPA}$$

其中子图位姿 $\Xi^m=\{\xi_i^m\}_{i=1,\ldots m}$ 和在世界中的扫描位姿 $\Xi^s=\{\xi_j^s\}_{j=1,\ldots n}$ 会被给出的一些约束优化。这些约束的形式与相对位姿 ξ_{ij} 和相关的协方差矩阵 Σ_{ij} 有关。对于一对子图 i 和扫描 j，姿态 ξ_{ij} 描述了在哪里子图的坐标扫描匹配。这个协方差矩阵 Σ_{ij} 是可以评估的，例如[15]中的方法，或局部使用带有 Ceres [14]估计特征（CS）的协方差。关于这样的约束的残余矩阵 E 由以下方程计算

$$E^2(\xi_i^m,\xi_j^s;\Sigma_{ij},\xi_{ij})=e(\xi_i^m,\xi_j^s;\xi_{ij})^T\Sigma_{ij}^{-1}e(\xi_i^m,\xi_j^s;\xi_{ij}), \quad (4)$$

$$e(\xi_i^m,\xi_j^s;\xi_{ij})=\xi_{ij}-\begin{pmatrix} R^{-1}(t_{\xi_i^m}-t_{\xi_j^s}) \\ t_{\xi_{i;\theta}^m}-t_{\xi_{j;\theta}^s} \end{pmatrix} \qquad (5)$$

损失函数 ρ，例如胡贝尔的损失，是用来减少在当扫描匹配增加了不正确的约束的优化—最优化问题时在（SPA）中可能出现的异常值的影响。例如，这可能发生在局部对称的环境中，如办公室。替代离群值的方法包括[16]。

B.分支边界扫描匹配

我们感兴趣的是最佳的像素精确匹配，

$$\xi^*=\arg\max_{\xi\in W}\sum_{k=1}^{K}M_{nearest}(T_\xi h_k) \qquad \text{(BBS)}$$

其中 W 是搜索窗口，$M_{nearest}$ 是将其参数舍入到最近的网格以扩展到所有的 R^2，这是将网格点的值扩展到对应像素。匹配的质量可以被进一步改进使用(CS)。

通过仔细选择步长，效率得到了提高。我们选择的角步长 δ_θ，保证扫描点最大范围 D 最大移动范围 $d_{\max}$ 不超过 R，即一像素的宽度。通过使用余弦公式，我们推出下列公式：

$$d_{\max} = \max_{k=1,...K} \| h_k \|, \quad (6)$$

$$\delta_\theta = \text{across}(1 - \frac{r^2}{2d_{\max}^2}) \quad (7)$$

当搜索窗口的尺寸为$Wx = Wy = 7\text{m}$和$W_\theta = 30^\circ$四，我们计算了覆盖它时所需要的完整步骤。

$$w_x = \left\lceil \frac{W_x}{r} \right\rceil, w_y = \left\lceil \frac{W_y}{r} \right\rceil, w_\theta = \left\lceil \frac{W_\theta}{\delta_\theta} \right\rceil \quad (8)$$

这引出了一个完整集 W ，它是以估计的位姿ξ_0为中心的搜索窗口。

$$\overline{\mathrm{W}} = \{-w_x,...,w_x\} \times \{-w_y,...,w_y\} \times \{-w_\theta,...,w_\theta\} \quad (9)$$

$$\mathrm{W} = \{\xi_0 + (rj_x, rj_y, \delta_\theta j_\theta) : (j_x, j_y, j_\theta) \in \overline{\mathrm{W}}\} \quad (10)$$

一个用来寻找的导航算法很容易被推导出来，见于算法 1，但对于我们已经知道的搜索窗口的尺寸，它可能会很慢。

算法 1：

Algorithm 1 Naive algorithm for (BBS)

```
best_score ← −∞
for j_x = −w_x to w_x do
  for j_y = −w_y to w_y do
    for j_θ = −w_θ to w_θ do
      score ← Σ_{k=1}^{K} M_nearest(T_{ξ0+(rj_x, rj_y, δ_θ j_θ)} h_k)
      if score > best_score then
        match ← ξ0 + (rj_x, rj_y, δ_θ j_θ)
        best_score ← score
      end if
    end for
  end for
end for
return best_score and match when set.
```

相反地，我们使用分支定界方法来有效计算在较大的搜索窗口中的ξ。参见算法 2 泛型算法。这种方法首先在混合整数线性规划的程序中提出[17]。关于这个主题的讲座是广泛的，参见作了简短概述[18]。

主要的思想字集的可能性表述为一棵树的结点。而树的根节点表示所有可能的解决方案，在我们的例子中即是 W。每个节点的子部分形成他们的父母的一个分区，因此他们一起代表同样的可能性。而叶节点为单个的，表示单一可行解。注意到算法是精确的。只要内部结点 c 的函数 score（c）是它的上一层边界，它就能提供和初级方法相同的解决方案。在这种情况下，每当节点是有界的，比已知最优解更好解到在这个子树并不存在。

为了得出一个具体的算法，我们必须决定节点选择、分支和计算上层边界的方法。

1）节点选择：在没有更好的情况下，我们的算法采用深度优先搜索（DFS）作为默认的选择。算法的效率取决于这棵树的大部分被修剪了。这取决于两个方面：一个好的上界，一个好的当前解决方案。后一部分由 DFS 帮助，它快速评估许多叶节点。既然我们不想加上差的的匹配来作为闭环的约束，我们便引入了一个当我们对最优解不感兴趣时的得分阈值。因为在实践中，门槛不会经常被超越，这减少了节点选择或查找初始启发式解的重要性。注意在 DFS 期间访问这些子结点的顺序，我们计算对每个子结点的得分上限，访问带有最大边界的最有希望的子节点。这方法便是算法 3。

2）分支规则：树中的每个节点都由一个元组整数 $c=(c_x,c_y,c_\theta,c_h)\in Z^4$ 表示。高度为 c_h 的节点将有多达 $2^{c_h}\times 2^{c_h}$ 种的可能的变换，但只代表以下特定旋转：

$$\overline{\overline{W}}_c=\left(\left\{(j_x,j_y)\in Z^2:\begin{matrix}c_x\le j_x<c_x+2^{c_h}\\ c_y\le j_y<c_y+2^{c_h}\end{matrix}\right\}\times\{c_\theta\}\right), \qquad (11)$$

$$\overline{W}_c=\overline{\overline{W}}_c\bigcap\overline{W} \qquad (12)$$

算法 2：

Algorithm 2 Generic branch and bound

```
best_score ← −∞
C ← C_0
while C ≠ ∅ do
  Select a node c ∈ C and remove it from the set.
  if c is a leaf node then
    if score(c) > best_score then
      solution ← n
      best_score ← score(c)
    end if
```

```
    else
      if score(c) > best_score then
        Branch: Split c into nodes C_c.
        C ← C ∪ C_c
      else
        Bound.
      end if
    end if
  end while
  return best_score and solution when set.
```

算法 3

Algorithm 3 DFS branch and bound scan matcher for (BBS)

```
best_score ← score_threshold
Compute and memorize a score for each element in C_0.
Initialize a stack C with C_0 sorted by score, the maximum
score at the top.
while C is not empty do
  Pop c from the stack C.
  if score(c) > best_score then
    if c is a leaf node then
      match ← ξ_c
      best_score ← score(c)
    else
      Branch: Split c into nodes C_c.
      Compute and memorize a score for each element
      in C_c.
      Push C_c onto the stack C, sorted by score, the
      maximum score last.
    end if
  end if
end while
return best_score and match when set.
```

叶节点高度为 $c_h = 0$ ，对应的解决方案为 $\xi_0 = \xi_0 + (\mathrm{rc_x}, \mathrm{rc_y}, \delta_\theta \mathrm{c}_\theta) \in \mathrm{W}$

在我们的算法 3 中，包含—传递所有可行解的根节点，，它并没有显式出现。

在固定的高度h_0中分成一组初始节点C_0以覆盖搜索窗口：

$$\begin{aligned}
&\overline{\mathrm{W}}_{0,x} = \{-w_x + 2^{h_0} + j_x : (j_x \in \mathrm{Z}, 0 \le 2^{h_0} j_x \le 2w_x)\},\\
&\overline{\mathrm{W}}_{0,\mathrm{y}} = \{-w_y + 2^{h_0} + j_y : (j_y \in \mathrm{Z}, 0 \le 2^{h_0} j_y \le 2w_y)\},\\
&\overline{\mathrm{W}}_{0,\theta} = \{j_\theta \in \mathrm{Z}, -w_\theta \le j_\theta \le w_\theta)\},\\
&C_0 = \overline{\mathrm{W}}_{0,x} \times \overline{\mathrm{W}}_{0,y} \times \overline{\mathrm{W}}_{0,\theta} \times \{h_0\}
\end{aligned} \tag{13}$$

在一个给定的$c_h > 1$ 的节点 c 上，我们分出了四个高度为$c_h - 1$ 的子节点

$$C_c = ((\{c_x, c_x + 2^{c_h - 1}\} \times \{c_y, c_y + 2^{c_h - 1}\} \times c_\theta) \bigcap \overline{\mathrm{W}}) \times \{c_h - 1\} \tag{14}$$

3）计算上界：剩余部分分枝定界法是一种有效的计算内部节点的上界方法，无论是在计算压力还是在边界的品质上。我们使用以下公式：

$$score(c) = \sum_{k=1}^{K} \max_{j \in \overline{\overline{W}}_{\mathrm{c}}} M_{nearest}(T_\xi, h_k) \tag{15}$$

为了能够高效地计算最大值，我们使用预先计算的网格$M^{c_h}_{precomp}$。预计算一个网格每个可能的高度c_h，使我们能够计算扫描点数的分数。注意，要能够做到这一点，我们也计算过超过$\overline{\overline{\mathrm{W}}}_{\mathrm{c}}$的最大值，在搜索空间的边界附近可能大于$\overline{\mathrm{W}}_c$。

$$score(c) = \sum_{k=1}^{K} \max_{j \in \overline{\overline{W}}_{\mathrm{c}}} M^{\mathrm{c_h}}_{\mathrm{pre}comp}(T_\xi, h_k) \tag{16}$$

$$M^{\mathrm{c_h}}_{\mathrm{pre}comp}(x, y) = \max_{\substack{x' \in |x, x + r(2^h - 1) \\ y' \in |y, y + r(2^h - 1)}} M_{nearest}(x', y') \tag{17}$$

注意，$M^h_{precomp}$有与M_{nearest}相同的像素结构，但这是在每个存储有最大值为$2^h \times 2^h$的像素中。图 3 是一个预先计算的网格的例子。

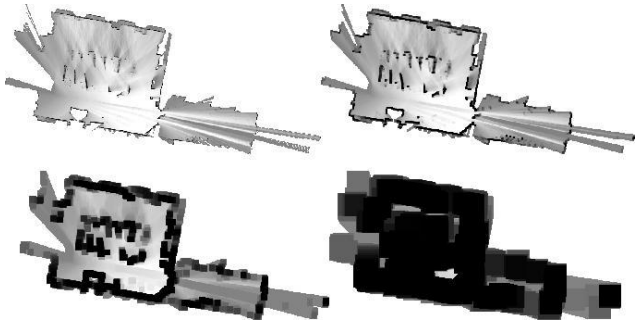

图 3 预想计算的尺寸分别为 1,4,16,64 的网格

为了保持预先计算网格的计算工作量在比较低的水平，我们一直等待直到概率网格不再接收进一步更新，然后我们计算一个预先计算网格的集合，并开始匹配它。

对于每一个预先计算的网格，我们计算的最大值为从每个像素开始的宽 2^h 像素行。使用这个—调解的结果，下一个预先计算的网格将会被构建。

当值被按照他们添加的顺序删除时，关于变化的值集合的最大值可以在均摊 O(l) 中保持最新。连续的最大值保存在一个可以递归地定义包含当前中所有值的最大值集合出现之后，依次列出所有的最大值的容器。对于一个空值集合，此列表为空。使用此计算网格的方法，可以计算出 O(n)，其中 n 是每个预先计算的网格的像素数。

计算上界的另一种方法去是计算更低分辨率的概率网格，可以逐次减半分辨率，见于[1]。由于额外的内存消耗是我们的方法可以接受的，所以我们不喜欢使用较低的分辨率概率网格，因为它将导致更差的的比（15）更差的边界，对性能会产生负面影响。

6. 实验结果

在本节中，我们给出了一些使用我们的 SLAM 算法的结果，它们是使用从传感器记录的数据计算出来的。首先，我们展示了利用在慕尼黑德意志博物馆使用背包传感器收集的数据计算的一些结果。第二，我们证明了我们的算法可以在廉价的硬件上实施，比如通过使用从吸尘器传感器中收集的数据，。最后，我们展示使用萝卜数据集[19]计算的结果，并与已经公开的结果进行比较。

A.现实世界的实验：德意志博物馆

通过使用在德意志博物馆收集的长达 1913s 或长达 2253m3s（根据计算解决方案）的传感器数据，我们计算了图 4 所示的图。在使用配有频率为 3.2GHz 的英特尔至强 E5-1650 的工作站，我们 SLAM 算法使用了 1018s 的 CPU 时间，使用了高达 2.2gb 内存和高达 4 个做闭环扫描匹配的后台线程。它在 360s 挂钟时间后完成，这意味着它实现了实时性能的 5.3 倍。

闭环优化的生成图由 11456 个节点和 35300 个边组成。优化问题（spa）每次都有几个节点运行。添加到图中。一个典型的解决方案需要大约 3 次迭代，在约 0.3s 完成。

实验结果如图 4 所示：

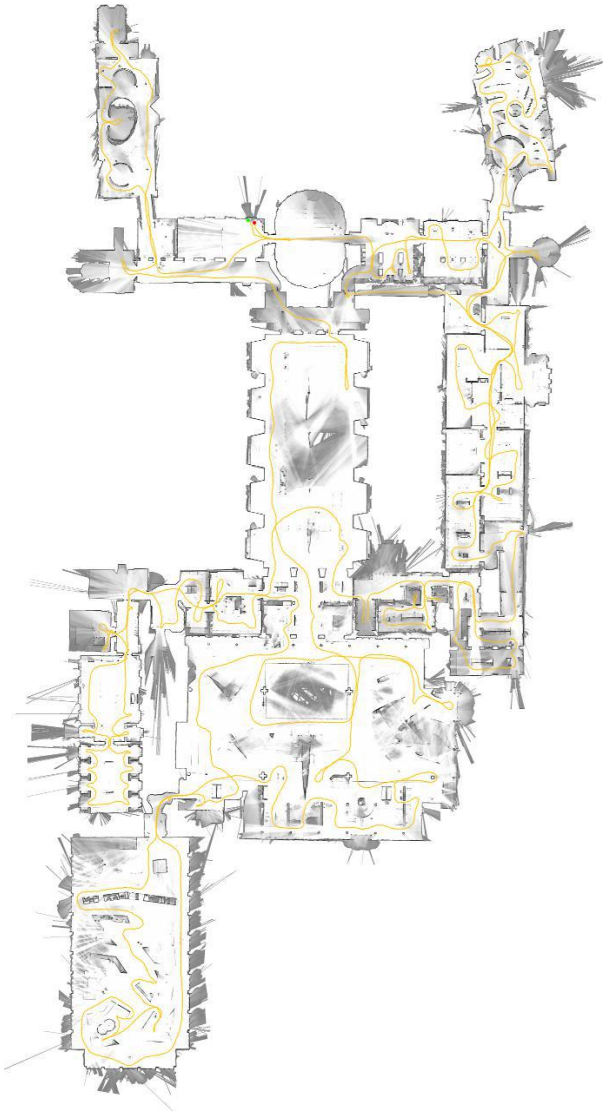

图 4.德意志博物馆实验结果图

B.现实世界的实验：Neato’s Revo LDS

Neato Robotics 公司在他们的吸尘器使用了一个叫做 Revo LDS[20]的激光测距传感器（LDS），传感器的成本低于 30 美元。我们把吸尘器放进货车，让它在调试连接的模式下以约 2 赫兹的频率扫描，我们从而获得数据。图 5 显示结果为 5cm 分辨率的平面图。为了评估平面图的质量，我们比较了由 5 条直线的激光卷尺计算得到的像素。结果见表一，所有值的单位为米。这些值大致处于预期的从线的每一端的到终点的大小顺序。

表 1

Laser Tape	Cartographer	Error (absolute)	Error (relative)
4:09	4:08	—0:01	-0:2 %
5:40	5:43	+0:03	+0:6 %
8:67	8:74	+0:07	+0:8 %
15:09	15:20	+0:11	+0:7 %
15:12	15:23	+0:11	+0:7 %

C.使用萝卜数据集进行比较

我们将我们的方法与在[21]中表述的使用基准度量的其他方法进行比较，比较了从相对位姿变到手动策划的真实关系间的误差。表二显示我们制图计算 SLAM 算法后的结果。

表 2

	Cartographer	GM
Aces		
Absolute translational	0.0375 ± 0.0426	0.044 ± 0.044
Squared translational	0.0032 ± 0.0285	0.004 ± 0.009
Absolute rotational	0.373 ± 0.469	0.4 ± 0.4
Squared rotational	0.359 ± 3.696	0.3 ± 0.8
Intel		
Absolute translational	0.0229 ± 0.0239	0.031 ± 0.026
Squared translational	0.0011 ± 0.0040	0.002 ± 0.004
Absolute rotational	0.453 ± 1.335	1.3 ± 4.7
Squared rotational	1.986 ± 23.988	24.0 ± 166.1
MIT Killian Court		
Absolute translational	0.0395 ± 0.0488	0.050 ± 0.056
Squared translational	0.0039 ± 0.0144	0.006 ± 0.029
Absolute rotational	0.352 ± 0.353	0.5 ± 0.5
Squared rotational	0.248 ± 0.610	0.9 ± 0.9
MIT CSAIL		
Absolute translational	0.0319 ± 0.0363	0.004 ± 0.009
Squared translational	0.0023 ± 0.0099	0.0001 ± 0.0005
Absolute rotational	0.369 ± 0.365	0.05 ± 0.08
Squared rotational	0.270 ± 0.637	0.01 ± 0.04
Freiburg bldg 79		
Absolute translational	0.0452 ± 0.0354	0.056 ± 0.042
Squared translational	0.0033 ± 0.0055	0.005 ± 0.011
Absolute rotational	0.538 ± 0.718	0.6 ± 0.6
Squared rotational	0.804 ± 3.627	0.7 ± 1.7
Freiburg hospital (local)		
Absolute translational	0.1078 ± 0.1943	0.143 ± 0.180
Squared translational	0.0494 ± 0.2831	0.053 ± 0.272
Absolute rotational	0.747 ± 2.047	0.9 ± 2.2
Squared rotational	4.745 ± 40.081	5.5 ± 46.2
Freiburg hospital (global)		
Absolute translational	5.2242 ± 6.6230	11.6 ± 11.9
Squared translational	71.0288 ± 267.7715	276.1 ± 516.5
Absolute rotational	3.341 ± 4.797	6.3 ± 6.2
Squared rotational	34.107 ± 127.227	77.2 ± 154.8

为了比较，我们引用结果图的映射（21）。此外，我们引述更多的表三中已公布了的从[21]中获得的结果。误差是以米和度单位形式给出的，以绝对值或平方表示，还有它们的标准差。

每个公开的数据集都用一个唯一的传感器收集。传感器的配置不同于我们这的背包。因此，需要采用各种不同的算法参数，以产生合理结果。根据我们的经验，只需调整统一制图器就能将算法与传感器配置匹配，而非与的特定环境匹配。因为每个公共数据集都有唯一的传感器配置，我们不能确定我们的参数能否适合特定的位置。唯一的例外是弗莱堡

医院数据集，它有两个独立的关系文件。我们使用局部关系来调整参数，但在全局关系上也看到了良好的结果。如表 3，4，5 所示：

表 3

	Cartographer	Graph FLIRT
Intel		
Absolute translational	0.0229 ± 0.0239	0.02 ± 0.02
Absolute rotational	0.453 ± 1.335	0.3 ± 0.3
Freiburg bldg 79		
Absolute translational	0.0452 ± 0.0354	0.06 ± 0.09
Absolute rotational	0.538 ± 0.718	0.8 ± 1.1
Freiburg hospital (local)		
Absolute translational	0.1078 ± 0.1943	0.18 ± 0.27
Absolute rotational	0.747 ± 2.047	0.9 ± 2.0
Freiburg hospital (global)		
Absolute translational	5.2242 ± 6.6230	8.3 ± 8.6
Absolute rotational	3.341 ± 4.797	5.0 ± 5.3

表 4 闭环检测准确性

Test case	No. of constraints	Precision
Aces	971	98.1 %
Intel	5786	97.2 %
MIT Killian Court	916	93.4 %
MIT CSAIL	1857	94.1 %
Freiburg bldg 79	412	99.8 %
Freiburg hospital	554	77.3 %

表 5 耗时表现

Test case	Data duration (s)	Wall clock (s)
Aces	1366	41
Intel	2691	179
MIT Killian Court	7678	190
MIT CSAIL	424	35
Freiburg bldg 79	1061	62
Freiburg hospital	4820	10

所有的数据集之间的最显著的差异是频率和激光扫描质量，以及可用性和里程计的质量。尽管在公共数据集使用相对过时的传感器硬件，SLAM 的表现一贯在我们的预期，甚至在 MIT CSAIL 的例子中，我们的表现比图映射差了很多。对于英特尔数据集，我们优于图形映射，但对于图形调弄却不是。在麻省理工学院 Killian Court 中，我们在所有的指标都超越图映射。在所有其他情况下，都优于图映射和图像调弄，但并不是在所有指标上

自从我们在子图和扫描之间加入了闭环约束，数据集就包含了不真实。也很难将数字与基于扫描到扫描的其他方法进行比较。表四显示了每个测试用例（真和假阳性）所添加的闭环约束的数量，以及精度，这是真阳性的一部分。我们决定把在计算最优化过程中没有违反 20cm 以上或 1 度的真阳性约束集作为所有闭环约束的子集。我们看到，当我们扫

描子图匹配过程会产生误报的情况，这些必须在最优化过程中处理，才能为所有测试用例提供足够数量的闭环约束。在最优化过程中使用的胡贝尔损失是使闭环鲁棒性异常的因素之一。在弗莱堡医院的情况中，选择低分辨率和低最低分数的闭环检测产生了相对较高的误报率。通过提高闭环检测的最小得分，可以提高精度，但这会降低在某些维度上的解的质量。作者认为，场地的实际情况是保证最终成图质量更好的基准。

SLAM 的参数没有调整以提高 CPU 性能。我们依然在表 V 提供墙上时钟时间，它们是在频率为 3.2GHz 的英特尔至强 E5-1650 的工作站上测量的。我们提供传感器数据的持续时间以进行比较。

7. 结论

这一研究已经通过在慕尼黑的德意志博物馆中实验得到了验证。作者在此感谢博物馆的行政部门对我们工作的支持。

我们也对使用手动验证的关系和通过使用 Robotics Data Set Repository (Radish) 数据计算所得到的结果进行了比较 。感谢 Patrick Beeson, Dieter Fox, Dirk Hähnel, Mike Bosse, John Leonard,Cyrill Stachniss 等提供了这一数据集。而弗莱堡医院的数据集是由 Bastian Steder,Rainer Kümmerle, Christian Dornhege, Michael Ruhnke,Cyrill Stachniss, Giorgio Grisetti, 和 Alexander Kleiner.所提供的。

8. 参考文献

[1] E. Olson, “M3RSM: Many-to-many multi-resolution scan matching,” in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), June 2015.

[2] K. Konolige, G. Grisetti, R. Kummerle,¨ W. Burgard, B. Limketkai, and R. Vincent, “Sparse pose adjustment for 2D mapping,” in IROS, Taipei, Taiwan, 10/2010 2010.

[3] F. Lu and E. Milios, “Globally consistent range scan alignment for environment mapping,” Autonomous robots, vol. 4, no. 4, pp. 333– 349, 1997.

[4] F. Mart´ın, R. Triebel, L. Moreno, and R. Siegwart, “Two different tools for three-dimensional mapping: DE-based scan matching and feature-based loop detection,” Robotica, vol. 32, no. 01, pp. 19–41, 2014.

[5] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, “A flexible and scalable SLAM system with full 3D motion estimation,” in Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR). IEEE, November 2011.

[6] M. Himstedt, J. Frost, S. Hellbach, H.-J. Bohme,¨ and E. Maehle, “Large scale place recognition in 2D LIDAR scans using geometrical landmark relations,” in Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on. IEEE, 2014, pp. 5030– 5035.

[7] K. Granstrom,¨ T. B. Schon,¨ J. I. Nieto, and F. T. Ramos, “Learning to close loops from range data,” The International Journal of Robotics Research, vol. 30, no. 14, pp. 1728–1754, 2011.

[8] G. D. Tipaldi, M. Braun, and K. O. Arras, “FLIRT: Interest regions for 2D range data with applications to robot navigation,” in Experimental Robotics. Springer, 2014, pp. 695–710.